U.S. Express Mail No.: 022537048 US
Filing Date: February 6, 2004

PATENT
Howrey Matter No. 02453.0020.NPUS00
Extreme Docket No.: P108

APPARATUS, METHOD AND SYSTEM FOR IMPROVING NETWORK SECURITY

INVENTORS

Shehzad T. Merchant
Derek H. Pitcher
Victor C. Lin
Manish M Rathi
Jia-Ru Li
Matthew R. Peters
Balaji Srinivasan
Vipin K. Jain
Amit K. Maitra

RELATED APPLICATIONS

This application is related to U.S. Patent Application No. To Be Determined, Howrey Docket No. 02453.0019.NPUS00, entitled "Method And System For Location-Based Access Control In A Computer Network"; U.S. Patent Application Serial No. To Be Determined, Howrey Docket No. 02453.0021.NPUS00, entitled "Unified Adaptive Network Architecture." Each of the foregoing applications is filed concurrently herewith, and owned in common by the assignee hereof. Moreover, each of these applications is fully incorporated herein by reference as though set forth in full.

FIELD OF THE INVENTION

This invention generally relates to computer networks, and more specifically, to a computer network that includes features for improving network security by limiting the amount and type of information stored at network edge devices.

BACKGROUND OF THE INVENTION

Computer security, particularly network security, has become a significant concern in recent years. Network security protocols and procedures are generally designed to prevent unauthorized access to computer networks and the information stored on computers connected to the networks. Without security features in place, unauthorized users (frequently referred to as hackers) can access a network and steal or corrupt stored information and otherwise disrupt operation of the network. Such information theft is of particular concern on networks that provide access to sensitive information, e.g., those networks used in enterprise environments, such as corporations or government entities.

One way to gain unauthorized access to a computer network is to steal a network device. Network devices (i.e., computers and other electronic devices that communicate information over the network) often store information and data that are necessary to access the network. In particular, these devices often store image and configuration information in their local non-volatile memories (e.g., hard drives, EEPROM, non-volatile RAM, or the like). An image includes a software program that is executed by the device in order to access and operate on the network, and the configuration information is data that permits the device (or clients on or attached to the device) to access the network. These data can include security keys, such as encryption and/or authentication keys, user IDs, passwords, addresses, and the like.

Local storage of image and configuration information can lead to significant security compromises of the network. If the device is stolen, the thief will not only be able to use the device on another network, but will also have access to the sensitive configuration information, which will potentially permit unauthorized access to the network.

The theft of network devices is of particular concern in contemporary wireless networks that have access points (APs), such as those employing an IEEE 802.11 (Wi-Fi) protocol. APs are network edge devices that allow end users to wirelessly connect their computers or other personal devices to local area networks (LANs). APs are frequently deployed in open areas with full public access, making them susceptible to theft.

Accordingly, there is a need for an improved design and approach to managing network devices that not only deters their theft, but also reduces the risk of unauthorized network use.

SUMMARY

It is an advantage of the present invention to provide a network device, system and method for improving network security and deterring theft of network edge devices. The aforementioned objectives are accomplished by keeping sensitive information, such as executable images and configuration information, within the network itself, instead of permanently storing the images and information locally at network devices, such as APs.

In accordance with an embodiment of the invention, a network system includes a network component for storing sensitive information and a network device that is not fully operational until it is attached to the network and has downloaded the information. When downloaded, the sensitive information is stored in the network device so that when the device is disconnected from the network, the information is erased, making the device inoperative and removing the sensitive information, such as passwords, user IDs, network security keys, and the like. Disabling the network device in this manner not only prevents the theft of sensitive information, but also discourages theft of the device because it cannot be operated on another network without the information.

According to one aspect of the invention, the network device can include a bootstrap program for downloading an executable image. Like the configuration information, the executable image is also stored so that it is likewise erased when the device is disconnected from the network. The executable image is necessary for the device to operate on the network, and it also generates a request to the network. In response to the request, a server on the network downloads the configuration information to the device, where it is stored in the volatile memory and permits the device to become fully operational. In accordance with another embodiment of the invention, a network system includes a switch having at least one port for attaching a device so that information can be communicated between the device and the network system. The

device is not fully operational when it is first connected to the port. The network system also includes application(s) and server(s) for downloading configuration information from the network to the device in response to a request from the device. The information is stored in a volatile memory in the device. The device becomes operable on the network after the configuration information is downloaded into the volatile memory.

In accordance with a further embodiment of the invention, a system includes a network and a network device. The network device includes a network interface, a memory whose contents are erased upon loss of power to the device, and a bootstrap program for downloading and storing an executable image in the memory. The device is not fully operational when it is first connected to the network. The network includes a switch having at least one port for connecting to the network interface so that information can be communicated between the device and the network. The network also includes server(s) and/or application(s) for downloading the executable image from the network into the memory in response to a request from the bootstrap program. The server(s) and application(s) also download configuration information from the network to the memory in response to a request generated by running the executable image on the network device. The network device becomes fully operational on the network after the configuration information is downloaded into its memory.

Method counterparts to these embodiments are also provided. Other embodiments, systems, methods, features and advantages of the invention will be or will become apparent to one with skill in the art upon examination of the following figures and detailed description. It is intended that all such additional embodiments, systems, methods, features and advantages be included within the scope of the invention, and be protected by the accompanying claims.

BRIEF DESCRIPTION OF THE DRAWINGS

The components in the figures are not necessarily to scale, emphasis instead being placed upon illustrating the principles of the invention. In the figures, like reference numerals designate corresponding parts throughout the different views.

FIG. 1 is a flowchart showing a method of protecting network information and deterring theft of networked devices in accordance with an embodiment of the invention.

FIG. 2 is a block diagram of an exemplary network system in accordance with another embodiment of the present invention.

FIG. 3 is a message flow diagram showing an exemplary power-up sequence for an edge device included in the system of FIG. 2.

FIGS. 4-10 illustrate exemplary formats for messages sent between the edge devices and the LAN switch included in the system of FIG. 2.

DETAILED DESCRIPTION

Turning now to the drawings, and in particular to FIG. 1, there is illustrated a flowchart 150 showing a method of protecting information in accordance with an embodiment of the invention. In step 152, sensitive network information, that is, information not generally available to the public and useful for accessing the network, e.g., configuration data and software images, are stored within a network. The information is stored on a network component, such as a switch, server or networked storage device, that is located in a secure environment of the network. The secure environment prevents unauthorized access to the network component. The secure environment can limit both physical access (e.g. the network component is placed in a locked room) and/or access by way of network communications (e.g., the network component is placed behind a firewall or other network security device).

In step 154, a network device is attached to the network. The network device can be any electronic device capable of communicating with the network. Generally, the network device can be placed in an unsecured environment, where it is susceptible to theft. For example, the network device is preferably a wireless AP. Prior to being attached to the network, the network device lacks the sensitive information stored on the network, and thus, cannot initially access all of the services provided by the network. Further, the network device is inoperative, at least in part, without the sensitive information.

In step 156, the sensitive information is downloaded to the network device when it is connected to the network. When downloaded, the information is stored in the network device so that when the device is disconnected from the network, the information is erased, making the device inoperative and removing the sensitive information, such as proprietary software, passwords, user IDs, network security keys, or the like. To accomplish this, the sensitive information is stored in a memory that is erased when the device is disconnected, or additionally/alternatively, when it loses power, for example, the information can be stored in a volatile memory (e.g. a dynamic RAM) included in the network device.

In a preferred embodiment, the network device is powered by the network cable using a conventional Power over Ethernet (POE) scheme. With PoE, when the device is disconnected from the network, it, as well as its volatile memory loses power and the sensitive information is erased.

In step 158, the network device becomes operational. The network device is not fully operational until it is attached to the network and has downloaded the sensitive information. The sensitive information is installed and/or stored on the device so that the device becomes operational and can access the network.

In step 160, the network device is disconnected from the network. The sensitive information is erased from the network device when it is disconnected from the network. This causes the device to become inoperative, at least in part. Disabling the network device in this manner not only prevents the theft of sensitive information, but also discourages theft of the device because it cannot be operated on another network without the information.

FIG. 2 shows an exemplary network system 10 in accordance with an embodiment of the present invention. The network system 10 protects sensitive information and deters edge device theft by keeping network configuration data and software within a network 11. Specifically, sensitive information resides in a switch 20 included in the network 11. The system 10 downloads the information to "dumb" edge devices 22 only when they are connected to the network 11. The edge devices 22 are dumb in the sense that they cannot fully operate on a network without the configuration data and software. The configuration data and software are erased from the edge devices 22 when they are disconnected from the network 11. By losing this operational data and

software, the edge devices 22 essentially become inoperative and cannot be used interchangeably on other networks.

The network system 10 includes the network 11 comprising at least a local area network (LAN) switch 20 connected to an authentication server 18. The system 10 also includes one or more network edge devices 22, such as wireless access points (APs) that are also connected to the LAN switch 20.

The network 11 can also include other networking components (not shown), such as gateways, routers, additional servers hosting a variety of different applications, as well as links to other networks, such as an enterprise intranet or the public Internet.

The network components 18,20 can communicate with one another using any suitable access mechanism, such as SNMP, telnet, TCP/IP, HTTP or the like. Any suitable data network can be used to connect the components 18,20, such as a conventional Ethernet network.

The network switch 20 preferably communicates with the edge devices 22 over an Ethernet network.

The edge devices 22 can be any electronic network devices capable of communicating with the LAN switch 20 and including the features of the invention as described herein. The exemplary system 10 illustrates the devices 22 as being wireless APs. The edge devices 22 can also be wired devices that communicate with end user devices 14 over wired networks (not shown) instead of wireless channels.

Each edge device 22 includes a network interface (NI) 23, a boot ROM (read only memory) 24, and a volatile memory 26. The network interface 23 can be any suitable component for permitting network communication between the LAN switch 20 and the device 22, and is preferably a commercially-available Ethernet card. The boot ROM 24 can be any programmable ROM device, such as a flash or EEPROM (electrically erasable programmable read only memory), for storing a bootstrap program at the device 22. The volatile memory can be any suitable type of memory device that is erased when the device 22 is disconnected from the LAN switch 20. For example, the volatile memory 26 can be a dynamic random access memory (DRAM), which is erased when it loses power.

The edge device can also include a central processing unit (CPU) (not shown), such as a commercially-available microprocessor, for executing the bootstrap program as well as any software images and applications downloaded from the network 11. The CPU can be connected to boot ROM 24, memory 26, and NI 23. It can also control the NI 23.

Where the edge device 22 is a wireless AP, it can provide network access to one or more wireless end user devices 14, such as personal computers, laptops, PDAs, phones or the like. The end user devices 14 and the AP can communicate using a conventional wireless protocol, such as a WiFi protocol based on one of the IEEE 802.11 standards.

The user devices 14 can also be voice over IP phones that can have access to the network 11 via the edge devices 22. The voice over IP phones can be configured to download the images and configuration data from the switch 20 and through the edge devices 22. The voice over IP phones can store the configuration data and images in volatile memory so that when they are powered down or lose the signal from the edge devices 22, the configuration data and images are erased.

Before being connected to the LAN switch 20, the edge devices 22 are not fully functional. They lack a runtime device image (software) and configuration information that is required for them to be fully operational. This software and information is stored on the switch 20 in the network 11 and downloaded to the devices 22 when they are connected to the switch 20. This bootup procedure is described more fully below, in connection with FIG. 3.

The LAN switch 20 is a CPU-based network switch having interface cards, memory, and a real-time kernel and operating system (OS) for executing one or more software applications to support networked Ethernet communications and to perform in accordance with the various aspects of the invention disclosed herein. The real-time kernel and operating system (RTOS) can be VxWorks®, available from Wind River, Inc. of Alameda,CA.

The LAN switch includes a conventional AgentX server 17, a trivial file transfer protocol (TFTP) server 16 and memory 33 for storing at least one access point (AP) software image 34, at least one AP bootrom image 36 and configuration data. The

memory 33 storing the images 34,36 and data is preferably a conventional compact flash card, such as part no. W7B6064MIXG-T, available from Wintec.

The AP software images 34,36 and AP configuration data can be bundled with self-extracting switch software. During initialization of the switch 20, the bundled software can be loaded onto the switch 20. The bundled software can be compressed and wrapped so that it is self-extracting using a commercially-available software utility. When the bundled software is loaded onto the switch, the images 34,36 and configuration data are self-extracted and stored in the flash memory.

The compact flash memory has FAT16 file system structure on it. Thus, the AP image 34, AP bootrom 36, as well as other images and data can be stored as separate files on the compact flash file system. Using standard programming techniques, the flash file system can be registered with the VxWorks OS so that the files can be accessed by the OS and applications, such as the TFTP server 16, using standard OS calls.

The LAN switch 20 includes one or more ports 21 corresponding to each of the network devices 22. The ports 21 permit network communications between the devices 22 and the network 11. The LAN switch 20 can be a LAN access concentrator (LAC), i.e., it can be the network connection point for plural edge devices 22. All communications between the network 11 and the devices 22 pass through the switch 20.

The TFTP server 16 permits edge devices 22 to access images 34, 36. The images and data are downloaded to the edge devices 22 during the power on bootup sequence, as described in further detail below with reference to FIG. 3. The TFTP server 16 can also store other information. The TFTP server 16 is based on RFC 1350 and includes all features suggested in RFC2347.

On the switch 20, the TFTP server 16 operates as follows. The TFTP daemon task is started when switch 20 powers up. A TFTP task is created to process requests from the edge devices 22 and elsewhere. The server 16 supports multiple instances, i.e., one daemon listens on port 69, and new tasks are created to handle each incoming request. The tftpd PUT is achieved by reading bytes from the file descriptor and writing bytes to the INET UDP socket. The tftpd GET will read data from the socket and write to a file descriptor.

For TFTP opcodes field, the server 16 supports five opcodes:

RRQ	read request
WRQ	write request
DATA	data packet
ACK	acknowledgement
ERROR	error code

For the options field, the server 16 uses the default value as specified in RFC 2347.

The format field of the TFTP request is translated. Specifically, the TFTP client on the edge devices 22 sends netascii and octet. The available formats for TFTP are netascii, ascii, octet, binary, image. The server 16 and clients translate the formats as follows:

"netascii" = "netascii"
"ascii" = "netascii"
"octet" = "octet"
"binary" = "octet"
"image" = "octet"

In an alternative architecture, the TFTP server 16 can be located elsewhere in the network 11, outside the LAN switch 20. Also, alternative protocols, other than TFTP, can be used to access and transfer the stored images. For example, BOOTP can be used as an alternative transfer mechanism for downloading the device images from the network 11 to the edge devices 22.

The switch 20 includes one or more applications for managing connections to the edge devices 22, as well as applications for downloading and updating edge device images and configuration information.

The switch 20 can be deployed in a secure wiring closet, which is not generally accessible to the public.

Any suitable network protocol can be used between the LAN switch 20 and the edge devices 22, such as a data packet scheme. Preferably Ethernet is used to communicate messages and information between the switch 20 and the devices 22.

The authentication server 18 can be a conventional RADIUS server that is configured to provide authentication services to the LAN switch 20. The server 18 can be used to authenticate an edge device 22 when it is first connected to the network 11, based on a device identifier stored in the device 22. The LAN switch 20 can include an authenticator operating in conformance with the IEEE 802.1x standard and can use the standard RADIUS protocol to transfer authentication requests and responses to and from the authentication server 18 via the switch 20. In addition, the server 18 can also authenticate end user devices 14 as they log onto the network 11. In an alternative architecture, the authentication server 18 can be included in the LAN switch 20 or located elsewhere in the network 11.

FIG. 3 is a message flow diagram 50 showing an exemplary power-up sequence for a dumb edge device 22 (e.g., an AP) included in the system 10 of FIG. 2. This bootup sequence illustrates an example method of protecting sensitive information by ensuring that it is only temporarily resident on the APs 22 while they are connected to the network 11.

In this bootup process, the AP 22 is first connected to a LAN switch 20 (e.g., a LAC) port using a network cable, such as a CAT-5 cable (step 52). Preferably, the AP 22 does not have an independent power source and takes power from the network cable using a convention scheme such as power over Ethernet (PoE). Using POE, the volatile memory 26, particularly any sensitive information contained therein, is erased when the AP 22 is disconnected from the switch 20. Upon receiving power, the AP 22 resets and starts its own bootup program that is stored in the boot ROM 24 (steps 54-56). When the AP 22 resets, its CPU points to a reset vector at the start of the bootup program. If the AP 22 ever loses power from network cable or connectivity from switch 20, it will restart the whole booting sequence again at the reset vector location.

While executing, the AP bootstrap program sends the AP Announce Message (FIG. 5) to the switch 20 (step 58) and waits for a response (step 60). If the response is not forthcoming, a retry will be used by the AP 22, if necessary, to avoid the loss of the Announce Message (step 62).

If authentication is successful, the switch 20 will decide the corresponding image(s) for the AP 22 to receive and send out an Announce Reply Message (FIG. 6).

Until it receives the Announce Message, the switch 20 waits (step 64). Upon receiving the Announce Message, the switch 20 determines whether a bootrom upgrade is necessary. (Step 66) If switch 20 determines that a bootrom upgrade is required from the contents of the Announce Message, the TFTP path and filename in the Reply Message is for the bootrom image (step 68), instead of the executable image (step 70). The switch 20 determines whether an upgrade is needed by comparing the bootrom version in the AP Announce Message with the version of the current bootrom program stored on the switch 20.

In the Announce Reply, the switch 20 delivers the switch IP address trivial file transfer protocol (TFTP) server identity, TFTP path/filename and other information that are needed for the AP 22 to download either the bootrom or AP image from the switch (step 72). The contents of this Reply message are discussed in more detail in connection with FIG. 6. The AP 22 then sends out the TFTP request (with retries, if necessary) to download the image and write it directly into AP's volatile memory 26 (e:g., DRAM) (steps 74 –76). The TFTP request is encapsulated in the packet format shown in FIG. 4.

In response to the TFTP request, the switch 20 sends one or more TFTP data packets to the AP 22 containing the requested image (step 78) and the goes into a "Wait TFTP_ACK" state (step 96). The TFTP data packets are encapsulated in the packet format shown in FIG. 4.

The new bootrom image is initially downloaded into the AP DRAM 26 and its checksum is computed (step 80) by the AP 22. If the checksum fails, the AP 22 resets and the bootup sequence restarts.

The AP 22 can verify the checksum of the image received from the TFTP server and send out a TFTP_ACK (FIG. 8) back to switch 20. When the switch 20 receives the ACK, the switch 20 transits to the "Wait for Image_ACK" state (step 98).

Next, the signature in the header is tested to see if it is the signature of a bootrom image or AP image (step 82). If it is the bootrom image, the AP 22 writes the bootrom image from the DRAM into the flash ROM, sends an acknowledge (ACK) to the switch 20

and then resets itself to start the process over, using the newly installed bootrom image to download the executable image (step 84). The ACK to the switch 20 can be sent using a keepalive message that identifies the AP 22 (FIG. 9).

If the image is the AP image, the AP 22 executes the newly downloaded image from the DRAM and ACKs the switch 20 (step 86). The ACK to the switch 20 can be sent using a keepalive message that identifies the AP 22.

Among other things, the image installs an operating system (OS) including a TCP/IP stack and AgentX client on the AP 22. The image uses the IP address, gateway and virtual LAN (VLAN) information in the Announce Reply Message to ready the TCP/IP stack. After this step, TCP/IP stack is ready and the AgentX protocol (based on RFC 2741) is used to deliver the configuration information from the AgentX server 17 located on the switch 20 to the AP DRAM 26.

In response to receiving the bootrom or AP image ACK, the switch 20 transits to the "Wait for AgentX" state (step 100).

After the TCP/IP stack is stabilized, AgentX is used to download the configuration information. The AP 22 initiates the process by sending an AgentX.Open request to the AgentX server running on the switch 20 (step 88). The switch 20 follows up by sending an AgentX.open_reply (step 92). This initial message sequence is then followed by one or more transactions of AgentX.Allocate/AgentX.Allocate_Reply and AgentX.Register/AgentX.Register_Reply between the AP 22 and switch 20 to agree on supported configuration parameters. (steps 90, 93) Once the parameter set has been agreed on a series of AgentX.Set messages are used to download the configuration information to the AP 22.

To close the session between the AP and the switch the AgentX.Close message is used. It can be initialized by either the switch 20 or AP 22.The switch 20 includes a state machine that is intended to remain synchronized with the AP 22 and uses timeout mechanism for corner cases. As shown in FIG. 3, steps 94-100 rely on the timeout mechanism to return the switch 20 to step 64 in the event of an unresponsive AP 22.

In addition, the switch 20 uses a counter called "Recv AP Announce" to decide if the switch 20 has received too many Announce Messages, for example, over thirty, at certain

points in the bootup sequence (steps 102-104). If the switch 20 receives too many AP Announce Messages before it completes the startup process, it disables the port (step 106). The counter will only incremented when switch 20 has time to process a request. When the switch 20 is highly loaded and does not have time to process Announce messages, incoming requests are collapsed and the counter is incremented once.

In a similar fashion, the switch 20 includes a counter called "Recv TFTP" to determine if the switch 20 has received too many TFTP requests from the AP 22 during the process (step 108).

In addition, the switch 20 can have timer between PoE up and the first AP Announce Message. If the switch 20 powers up the AP 22 and does not see the first Announce Message within a predetermined time period, it is assumed that there is something wrong with the AP 22 or the connection, and the switch 20 can power cycle the AP 22 again in an attempt to correct the situation.

The switch 20 configures the radio frequency (RF) profile of the AP 22 according to the IP address and gateway information specified in the Reply Message. (When the switch 20 receives the Announce Message, it replies with the Reply Message, which includes the IP settings). After the Renew_Config message is sent and the IP stack is reset the AP re-opens the AgentX session with the Switch 20. This includes sending the AgentX.Open, Alloc and Registration messages. In this case no AgentX.Set messages are sent, as configuration data has remained the same. FIGS. 4-9 illustrate exemplary formats for messages sent between the edge devices 22 and the switch 20 during the bootup procedure of FIG. 3. The messages are sent using Ethernet.

FIG. 4 shows the message format for encapsulating payload messages for transport between the AP 22 and switch over Ethernet., Each payload is encapsulated in an LLC+SNAP header. The header include source address (SA), destination address (DA), the LLC value, SNAP value, and protocol ID., version number, length, checksum, sequence number, and MAC address. The MAC address in the header is the sender's MAC address. The sequence number will be incremented by the sender for every packet sent.

FIGS. 5-9 show details of the message payloads carried in the packet format of FIG. 4 for the messages sent between the switch 20 and AP 22 during the bootup sequence discussed in connection with FIG. 3.

FIG. 5 shows an example of an AP Announce Message. The message contains the message length, AP hardware version (hardware ID, i.e., hardware major and minor revision IDs, and build number), bootrom image version (bootrom major and minor revision IDs, and build number), serial number, AP MAC address, hardware post status, and hardware post OK field. The hardware ID is a 16 byte ASCII string. The edge device can send this message every second, until Announce_Reply packet is returned by the LAN switch.

FIG. 6 shows an example of an Announce Reply Message. The message contains the IP address, subnet mask, default gateway and the management VLAN information on which the AP 22 should send the TFTP request. It also provides the "image_type" flag to identify whether the included URL is for the bootrom or AP image. The URL starts with "tftp://" and follows by the IP address of the TFTP server. The filename (or path name) follows. The fields are individually described as follows:

AP IP Address: IP address for the access point.

AP IP mask: mask for AP. For example, 24 means 255.255.255.0

AP default gateway Address: default gateway for the access point.

Switch IP Address: The LAN Switch IP to which the AP is connected.

AgentX port: After booting up, AP can start AgentX's TCP connection to this port.

Physical SW Port if_index: The physical port number on LAN switch that AP is connected to. AgentX can report SNMP trap using this information.

USE_VLAN_TAG: 0: do not use VLAN tag, 1: use VLAN tag.

VLAN Tag: Insert the tag into packets if Use_VLAN_TAG is set to 1.

Keep-alive timer: Timer for AP_KEEP_ALIVE in seconds.

Image type: 0: Boot code, 1:AP runtime image.

Boot URL can be, e.g., tftp://10.0.0.1/ Altitude-1.z. It can be a null terminated string.

FIG. 7 shows an example of a Renew_Config Message. The message contains a length, an AP IP address, AP IP mask, VLAN tag, default gateway, switch IP address, agentX port, and physical SW port index, and the other information shown for the AP 22 to

use. If no IP is assigned, the current AP IP will be used. If the IP address is specified, then any AgentX connection is shut, the AP IP stack is re-initialized, and AgentX is re-established. Having a new IP in this message usual means that the management VLAN changes. Re-initializing the AP IP stack will move the AP to new management VLAN.

FIG. 8 shows an example of a TFTP_ACK Message. The message contains status information indicating that the TFTP transfer of the bootrom or AP image is complete. The fields are defined as follows:

Image type: 0 - Boot ROM. 1 – AP

Status: 0: OK, 1: Fail

Retry #: 1 – 3 number of the TFTP retries.

FIG. 9 shows an example of a keepalive message from the AP 22 to the switch 20. The packet includes information identifying the AP 22. In response to the packet, the switch 20 updates its internal state. The AP 22 sends outs a keepalive packet every second. If three packets are missed, or if different AP 22 is identified by the packet, i.e., one that the switch 20 does not already associated with the port 21, then switch 20 resets the AP 22.

FIG. 10 shows an example of a reset message sent from the switch 20 to the AP 22. This type of packet instructs the AP 22 to reset. In response to receiving this message, the AP 22 does a soft reset of all interfaces, and its internal software stacks. The message can be sent by an administrator. After the switch 20 sends the reset, it starts a watchdog timer to track the boot-up sequence as its starts over. In the other word, the switch 20 waits for the Announce message and reply with Announce_Reply message. However, if the timer goes off before an Announce message is received, the switch concludes that the AP 22 has not reset itself successfully, and the switch 20 then power cycles the AP 22 through PoE.

While various embodiments of the invention have been described, it will be apparent to those of ordinary skill in the art that many more embodiments and implementations are possible that are within the scope of this invention. For example, any combination of any of the systems or methods described in this disclosure are possible.

What is claimed is: